CHunt_Assignment2_PS1_PS2

Christophe Hunt

February 8, 2017

Problem Set 1

1. Show that $A^T A \neq AA^T$ in general. (Proof and demonstration.)

For a 2x3 matrix A:

```
A <- matrix(c(1,2,3,4,5,6), nrow=2, ncol=3)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

The inverse of this matrix is:

```
t(A)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
```

The product of the inverse of the matrix and the matrix is :

```
t(A)%*%A
```

```
##      [,1] [,2] [,3]
## [1,]    5   11   17
## [2,]   11   25   39
## [3,]   17   39   61
```

Whereas, the product of the matrix and the inverse is :

```
A%*%t(A)
```

```
##      [,1] [,2]
## [1,]   35   44
## [2,]   44   56
```

Therefore, $A^T A \neq AA^T$.

2. For a special type of square matrix A, we get $A^T A = AA^T$. Under what conditions could this be true? (Hint: The Identity matrix I is an example of such a matrix).

The conditions under which these are true are for special square matrix. In these instances, the inverse of the matrix is equal to the matrix $A^T = A$. Therefore, if $A^T = A$ then $A^T A = AA^T$.

Such as the matrix :

```
A <- matrix(c(1,2,3,2,1,2,3,2,1), nrow=3, ncol=3)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    2    1    2
## [3,]    3    2    1
```

The inverse of this matrix is:

```
t(A)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    2    1    2
## [3,]    3    2    1
```

The product of the inverse of the matrix and the matrix is :

```
t_AA <- t(A)%*%A
t_AA
```

```
##      [,1] [,2] [,3]
## [1,]   14   10   10
## [2,]   10    9   10
## [3,]   10   10   14
```

The product of the matrix and the inverse is :

```
A_At <- A%*%t(A)
A_At
```

```
##      [,1] [,2] [,3]
## [1,]   14   10   10
## [2,]   10    9   10
## [3,]   10   10   14
```

```
A_At == t_AA
```

```
##      [,1] [,2] [,3]
## [1,] TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE
```

Problem Set 2

Matrix factorization is a very important problem. There are supercomputers built just to do matrix factorizations. Every second you are on an airplane, matrices are being factorized. Radars that track flights use a technique called Kalman filtering. At the heart of Kalman Filtering is a Matrix Factorization operation. Kalman Filters are solving linear systems of equations when they track your flight using radars.

Write an R function to factorize a square matrix A into LU or LDU, whichever you prefer.

```
factorization <- function(M){
  if(nrow(M) == 2){
     if(all(M[, 1] == 0)){
           return(print("cannot solve"))
          }
     while (M[1,1] == 0){
            M <- M[c(2,1), ]
          }
         E21 <- diag(nrow(M))
         E21[2,1] <- -M[2,1] / M[1,1]
         U <- E21 %*% M
         L <- solve(U)
         print(U)
         print(L)
      } else {
         if(all(M[, 1] == 0)){
           return(print("cannot solve"))
          }
         while (M[1,1] == 0){
            M <- M[c(2,3:nrow(M),1), ]
         }
         E21 <- diag(nrow(M))
         E21[2,1] <- -M[2,1] / M[1,1]
         M2 <- E21 %*% M
         E31 <- diag(nrow(M))
         E31[3,1] <- -M2[3,1] / M2[1,1]
         M3 <- E31 %*% M2
         E32 <- diag(nrow(M))
         E32[3,2] <- -M3[3,2] / M3[2,2]
         U <- E32 %*% M3
         L <- solve(E21) %*% solve(E31) %*% solve(E32)
         return(list("U" = U, "L" = L))

         }
      }
```

```
A <- matrix(c(1,2,3,1,1,1,2,0,1),nrow=3)
factored <- factorization(A)
```

```
factored$U
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    2
## [2,]    0   -1   -4
## [3,]    0    0    3
```

```
factored$L
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    2    1    0
## [3,]    3    2    1
```